The main difference between the two different printf outputs is that one way occupies 16 bytes while the other occupies 10 bytes. This is due to something called padding. A computer reads from and writes to memory addresses in word intervals (meaning 2 bytes at a time). In doing so means the data is aligned, it is organized in such a way that addresses of data in memory can be accessed by some multiple of a word size. Since chars, ints, floats, and other data types require different bit sizes, we require padding to maintain this data alignment. By using packed attribute, the compiler packs the fields together and disregards padding. A char is 1 byte and a float is 4 bytes, since we have 2 chars and 2 floats, we expect to have 2*1 + 2*4 = 10 total bytes, which was our output after including packed attribute. Without it, we have a total of 16 bytes. Notice that 16 bytes is a multiple of a word size, which proves that padding is being incorporated without packed attribute.

The conclusion we can draw is that in a scenario where memory is plentiful, padding can be beneficial since it is easier to access various addresses of memory. Since data is organized and padded, we can consistently use multiples of a word size to reach specific data. However, in the scenario of our SJ2 board memory is not a luxury, we have to conserve as much as possible. Therefore disregarding padding and packing fields closely would be a great benefit for us.